# Cours INFO 3114 - HIVER 2018 - Travail Pratique 1

## Sujet : Disque dur et les bassins de tampon

Le premier travail pratique consiste à écrire un programme C++ un programme **simulant** le fonctionnement d'un bassin de tampon, la mémoire virtuelle et le temps d'accès au disque dur.

## Consignes générales

- Le travail est un **travail individuel.**
- Le travail serait fait en C++ avec le compilateur de **Visual Studio.**
- Vous devez remettre un fichier compressé **.zip** contenant:
  - **(1) les fichiers sources requis pour compiler votre programme (.cpp, .h, …)**
  - **(2) l'exécutable**
- Le travail doit être remis au plus tard le **vendredi 26 janvier 2018 à 17h** dans la **boîte de dépôt « TP1 »** par le système CLIC (http://clic.umoncton.ca).
- Aucun retard ne sera accepté.
- Si un travail est plagié en totalité ou en partie, la note sera zéro.

## Spécifications

Le travail consiste à écrire un programme qui s'exécutera dans la console.

Ce programme effectuera une **simulation** pour illustrer le fonctionnement d'un bassin de tampon pour réduire les accès disque vers un fichier. Puisque ce travail est une simulation, vous n'utiliserez pas un vrai fichier ou le vrai accès au disque dur. Toute votre simulation se déroulera en mémoire vive. La raison est que le but du travail est avant tout de comprendre le fonctionnement d'un bassin de tampons et voir la différence entre les heuristiques.

Vous **simulerez l'exécution de programmes en mémoire virtuelle** avec :

**Mémoire Secondaire**

Un disque dur comme mémoire secondaire, à un seul plateau, 4 pistes contenant 5 secteurs chacune. Chaque secteur contiendra des valeurs allant de 0 à 19 représentant des fichiers qui seront utilisés par des programmes.

int secteurs[4][5];

Pour l'utilisation du disque dur, nous allons utiliser les caractéristiques suivantes :

- Le disque dur a un plateau et quatre pistes divisées en cinq secteurs
- Le temps de rotation est de 10ms. On assume qu'il faut exactement une rotation pour positionner la tête.
- Le temps pour lire un secteur 2ms.
- Le temps pour lire une piste adjacente est de 2ms

**Mémoire principale**

Dans notre simulation, nous allons supposer que chacun des secteurs du disque dur est équivalent à un bloc de mémoire vive (virtuelle). Notre mémoire virtuelle a **5 blocs**.

int memoireVirtuelle[5];

**Les Programmes simulés**

Un programme est représenté par un tableau d'entiers de taille 10000, qui est rempli avec des nombres aléatoires entre 0 et 19. Ces nombres représentent les fichiers du disque dur. Vous devrez créer 3 de ces programmes.

instruction programme1[10000];
instruction programme2[10000];
instruction programme3[10000];

Une instruction est simplement une structure contenant une opération read ou write ou modify ainsi qu'un # de fichier 0 à 19
Struct instruction
Char op;  // 'r' ou 'w' ou 'm'
Int file;  // # de fichier, de 0..19

**Le bassin de tampons**
Vous créerez **un bassin de tampons**. Dans votre code, ce bassin prendra la forme d'une classe C++.

Le bassin de tampons possédera les caractéristiques suivantes :

- Il aura **5** tampons (5 blocs mémoire).
- Le bassin utilisera
    - L'heuristique 1 : « **Utilisation d'une file, premier entré, premier sortie (FIFO)** »
    - L'heuristique 2 : « **retirer l'information la moins souvent utilisée** ».
- Il utilisera le *passage par tampons*. Il devra y avoir les méthodes suivantes :
    - *void readFile(int file) :* placer un fichier dans un tampon.
    - *void modifyFile(int file) : L'utilisateur a changé le contenu d'un fichier, donc il faut changer le bit sale. Si le fichier ne se trouve pas déjà dans un tampon, alors il faut invoquer readFile(file)*
    - *void writeFile(int file)* : Sauvegarde un fichier sur le disque. S'il n'est pas dans un tampon, alors il n'a pas été modifié.
    - *void closeFile(int file)* : forcer l'écriture de ce fichier sur le disque dur.
    - *Void closeBuffer()* : Vide le bassin de tampons et force l'écriture des tampons dont le bit sale sont TRUE.
    - *int fileInBuffer(int file) : retourne -1 si le fichier ne se trouve pas dans un tampon, sinon retourne le # de bloc ou il se trouve.*
    - *void markDirty(int block) :* changer la valeur de « bit sale » à **TRUE** pour un bloc.

D'autre fonctions et structures devront être développées afin de faire les calculs nécessaires et assurer le bon fonctionnement comme l'initialisation et l'affichage du programme.

Votre programme s'appellera « **TP1-2018_H1.exe »** (pour l'heuristique 1) et « **TP1-2018_H2.exe** » (pour l'heuristique 2). Vos programmes s'exécuteront de la façon suivante :

1. Le programme affichera tout d'abord votre nom, prénom ainsi que votre code NI et l'heuristique utilisée.
2. Le programme affichera l'état du bassin de tampons dans la console (pour chaque tampon : le contenu la valeur du bit sale, …) et le contenu des programmes simulés.
3. Le programme demandera ensuite à l'utilisateur de choisir le programme (1 à 3) à exécuter**.**
4. Le programme effectuera l'opération choisie par l'utilisateur.
5. Le programme affichera les statistiques d'exécution avec et sans les tampons, ainsi que le pourcentage d'efficacité gagné.
6. Ensuite le programme retournera à l'étape 2.

**Important** : Pour ce travail, il est important que votre programme affiche dans la console toute l'information pertinente pour qu'un observateur puisse vérifier que votre simulation se déroule correctement sans avoir à regarder le code source de votre programme.
**Note** : Vous pouvez inclure les deux heuristiques dans un programme si vous le voulez, pourvu que vous laissiez l'utilisateur le choix de l'heuristique pour la simulation.

## Évaluation

Ce travail sera noté sur 100 points et comptera pour **7.5 %** de la note globale du cours.

- **10 point** pour un code source contenant suffisamment de commentaires, un code source lisible et bien organisé et que toutes les consignes soient respectées (incluant les consignes de remise du travail),
- **10 point** pour la clarté de l'affichage et la facilité à comprendre l'exécution de votre programme.
- **35 points** si le fonctionnement de l'heuristique 1 est correct.
- **35 points** si le fonctionnement de l'heuristique 2 est correct.
- **10 point** si l'information du temps d'utilisation du disque dur est correcte.

Bon travail !